EECS 591
Distributed Systems

Manos Kapritsos
Fall 2021

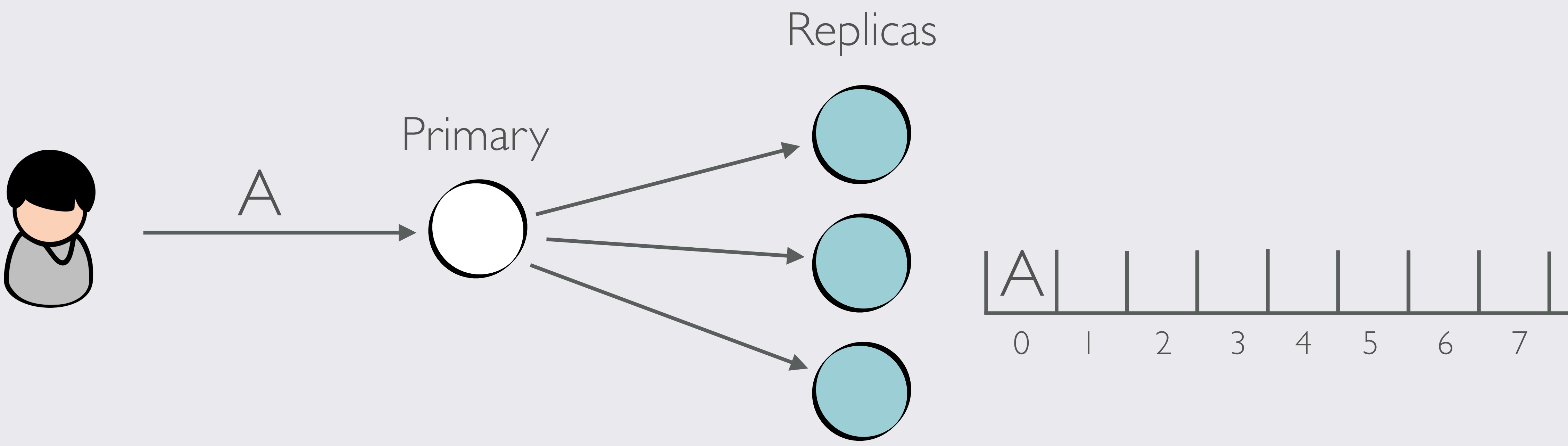
PBFT: A BYZANTINE RENAISSANCE

Practical Byzantine Fault Tolerance

(Castro, Liskov 1999-2000)

- First practical protocol for **asynchronous BFT replication**
- Like Paxos, PBFT is safe all the time, and live during periods of synchrony

The general idea



- One primary, 3f replicas
- Execution proceeds as a sequence of **views**
  - A view is a configuration with a well-defined primary
- Client sends signed commands to primary of current view
- Primary assigns sequence number to client’s command
- Primary is responsible for the command eventually being decided

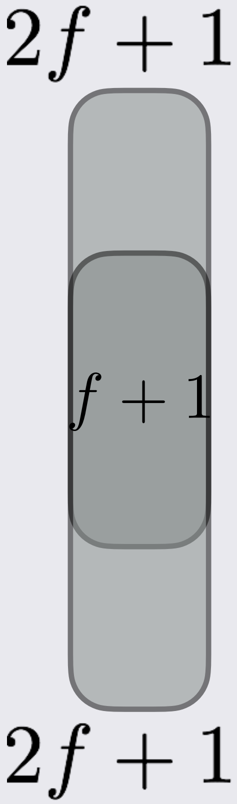
CERTIFICATES

Protocol steps are justified by **certificates**

- Sets (quorums) of signed messages from distinct replicas proving that a property holds

Certificates are of size at least $2f+1$

- Any two quorums intersect in at least **one correct** replica (for safety)
- There is always a quorum of correct replicas (for liveness)



PBFT: Normal operation

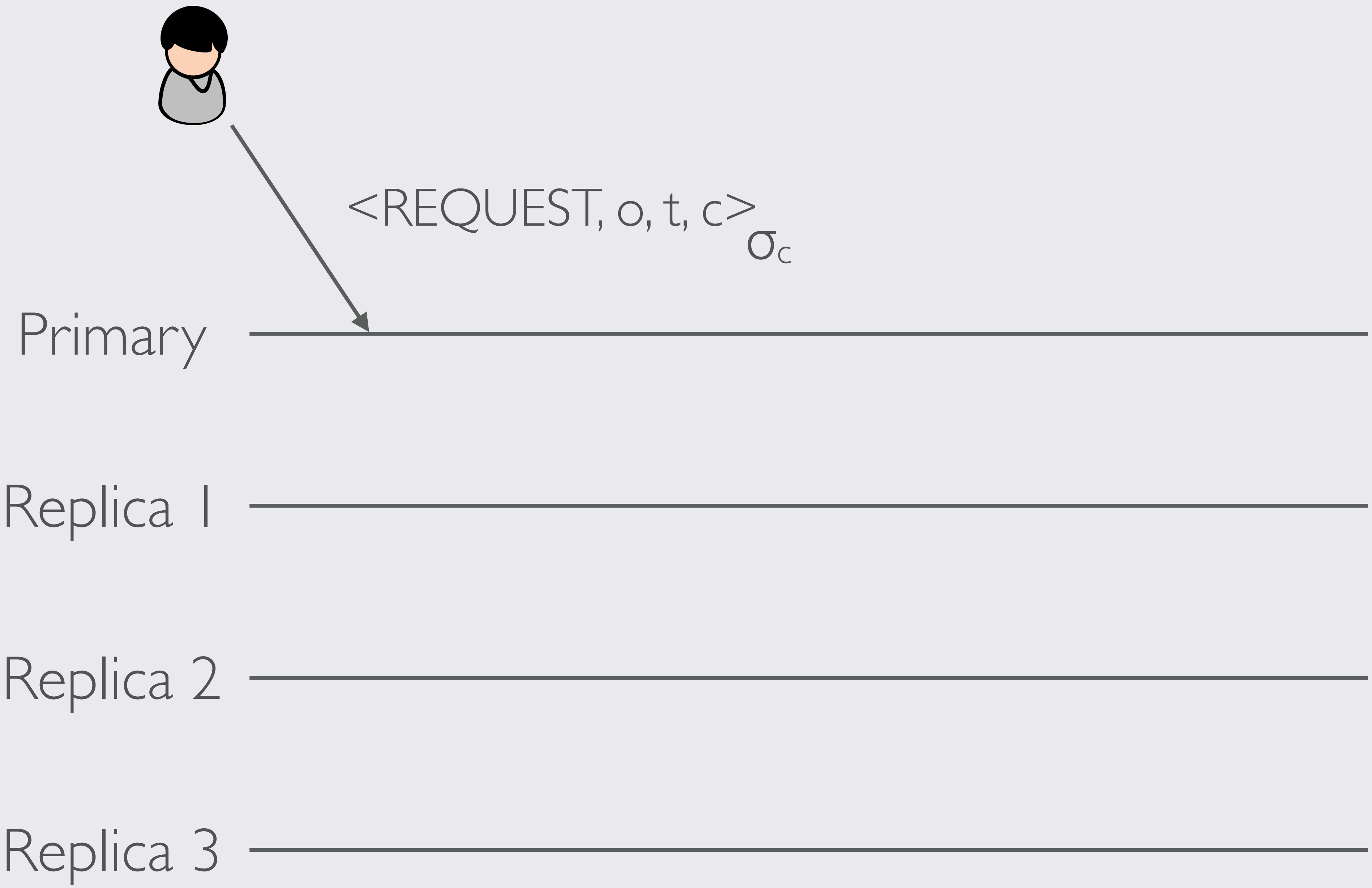
Three phases:

- **Pre-prepare** assigns sequence number to request
- **Prepare** ensures consistent ordering of requests within views
- **Commit** ensures consistent ordering of requests across views

Each replica maintains the following state:

- Service state
- A **message log** with all messages sent or received
- An integer representing the replica’s current view

Client issues request



<REQUEST, o, t, c>$_{\sigma_c}$

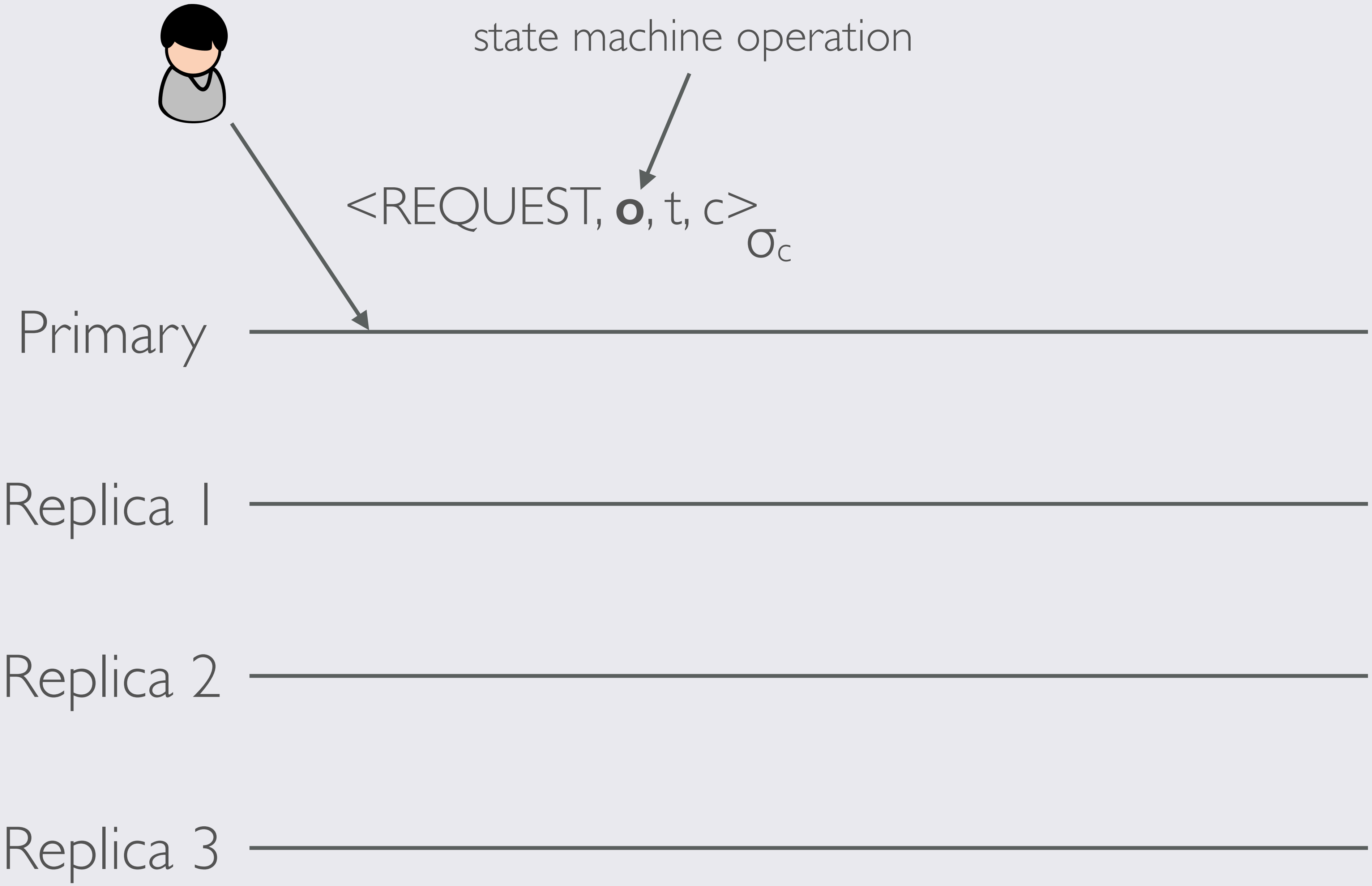
Primary

Replica 1

Replica 2

Replica 3

Client issues request



state machine operation

$\langle$REQUEST, **o**, t, c$\rangle_{\sigma_c}$

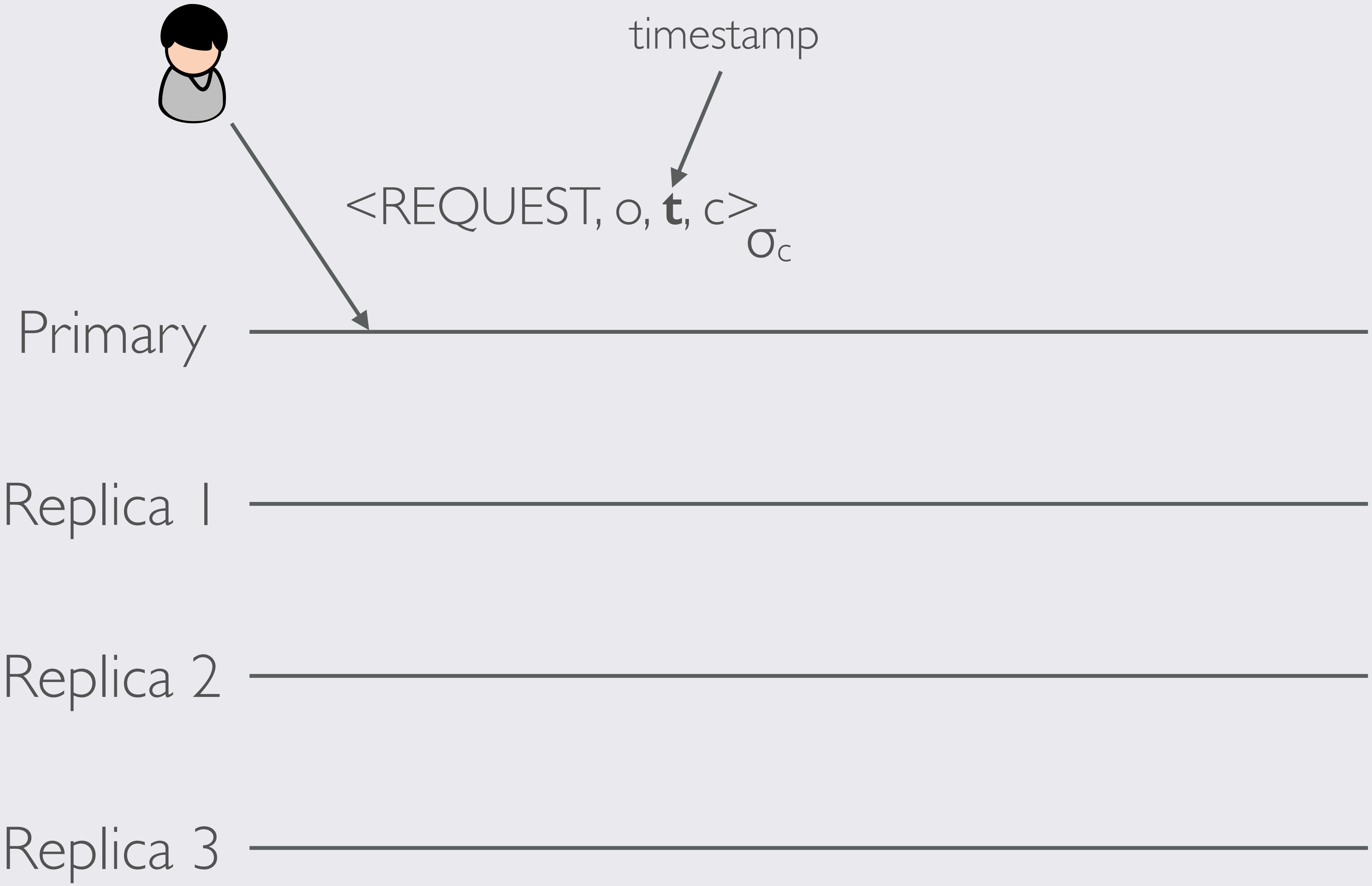
Primary

Replica 1

Replica 2

Replica 3

Client issues request



timestamp

<REQUEST, o, **t**, c>$_{\sigma_c}$

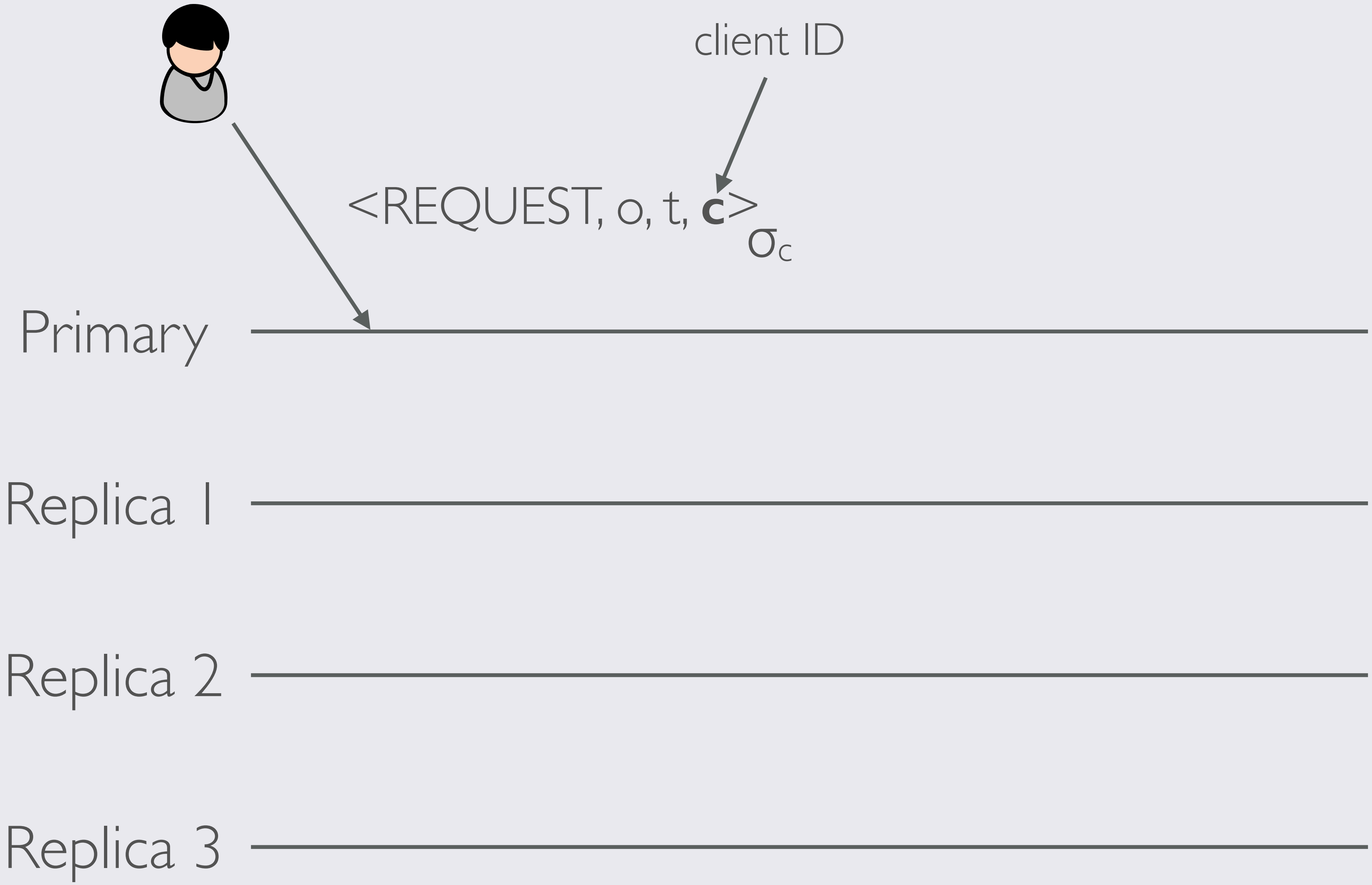
Primary

Replica 1

Replica 2

Replica 3

Client issues request



client ID

<REQUEST, o, t, **c**>$_{\sigma_c}$

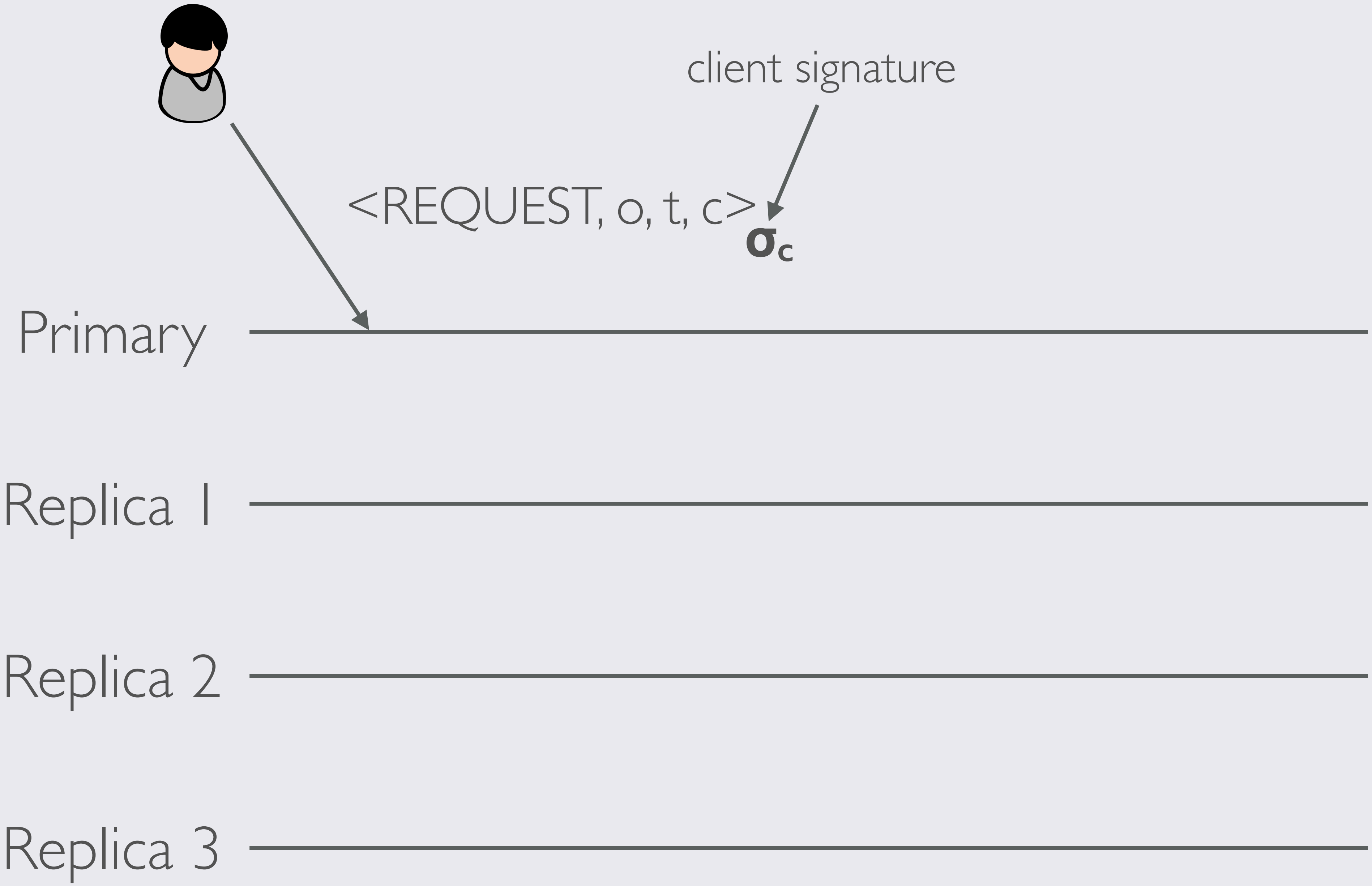
Primary

Replica 1

Replica 2

Replica 3

Client issues request



client signature

<REQUEST, o, t, c>$_{\sigma_c}$

Primary

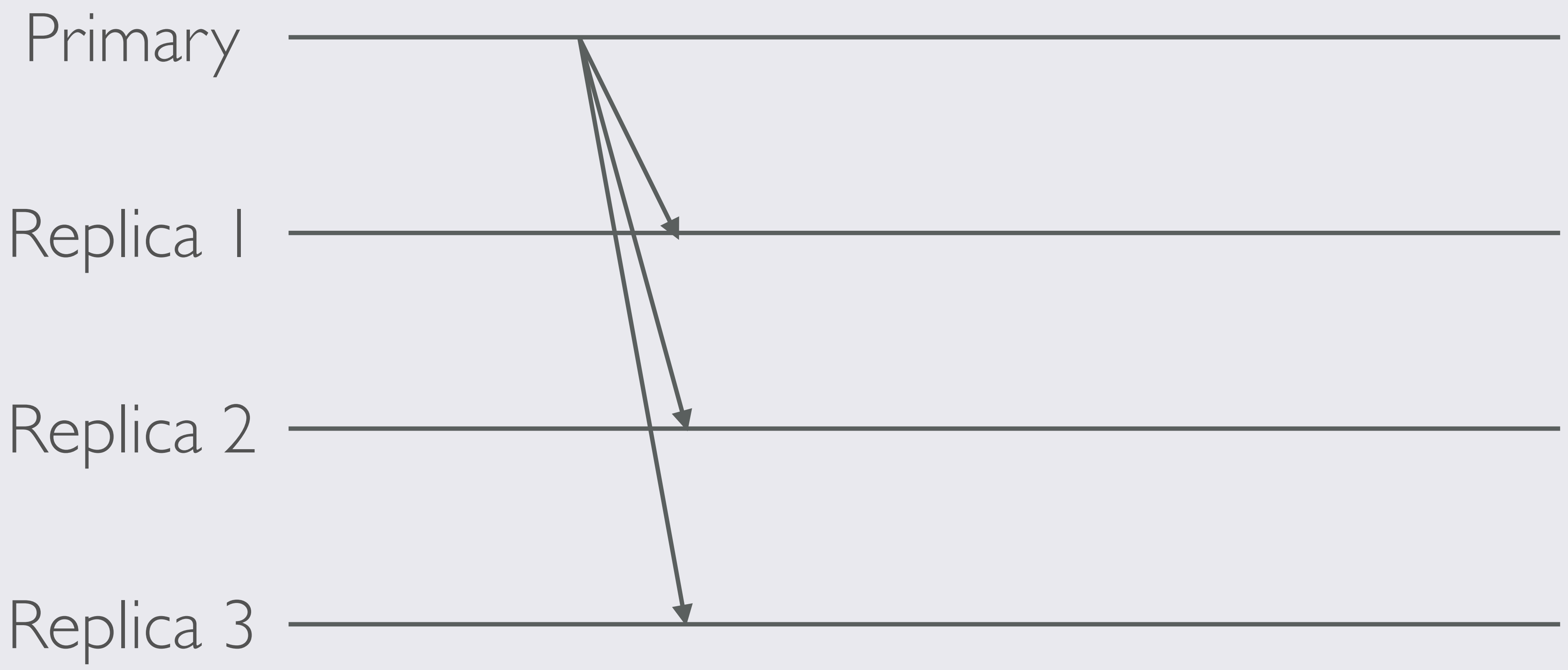
Replica 1

Replica 2

Replica 3

PRE-PREPARE

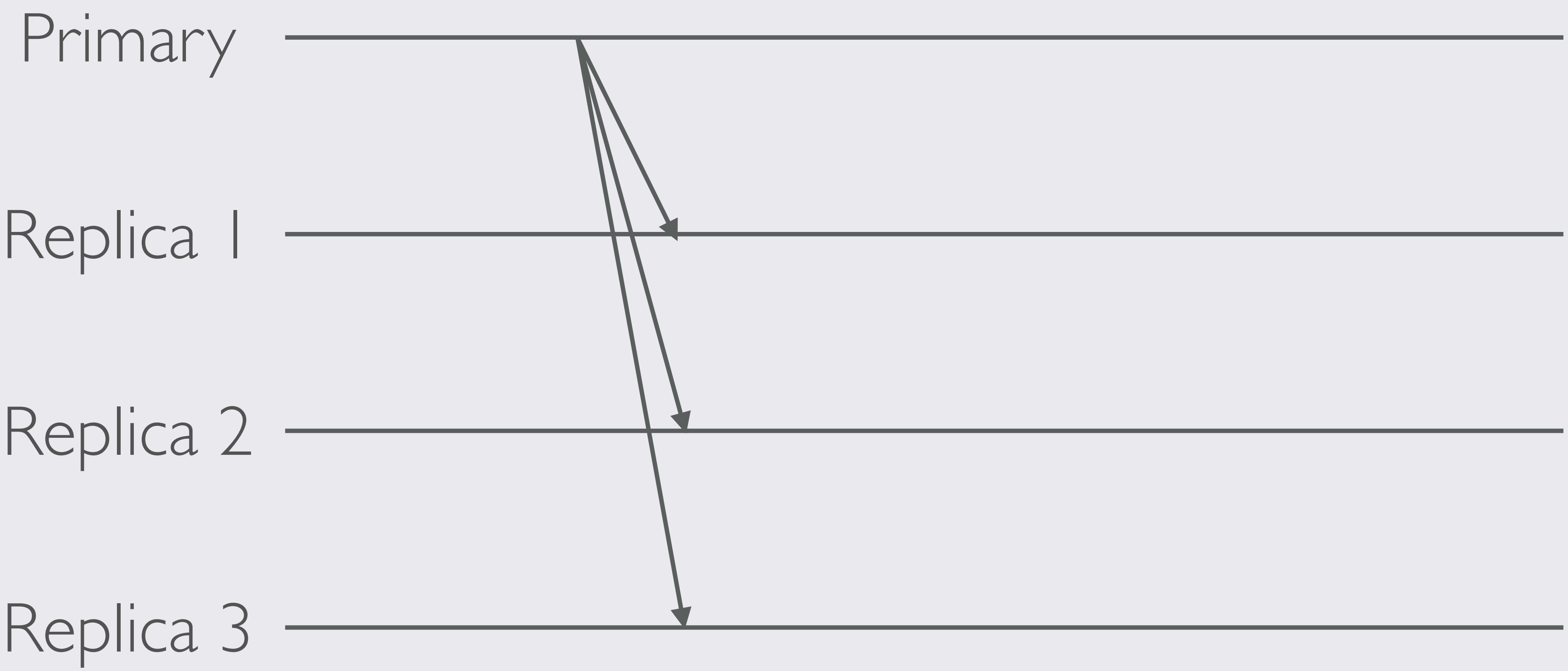
Primary sends <<PRE-PREPARE, v, n, d>$_{\sigma_p}$, m> to all replicas



PRE-PREPARE

current view

Primary sends <<PRE-PREPARE, **v**, n, d>$_{\sigma_p}$, m> to all replicas



Primary

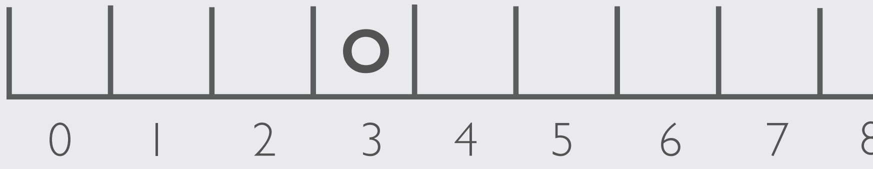
Replica 1

Replica 2

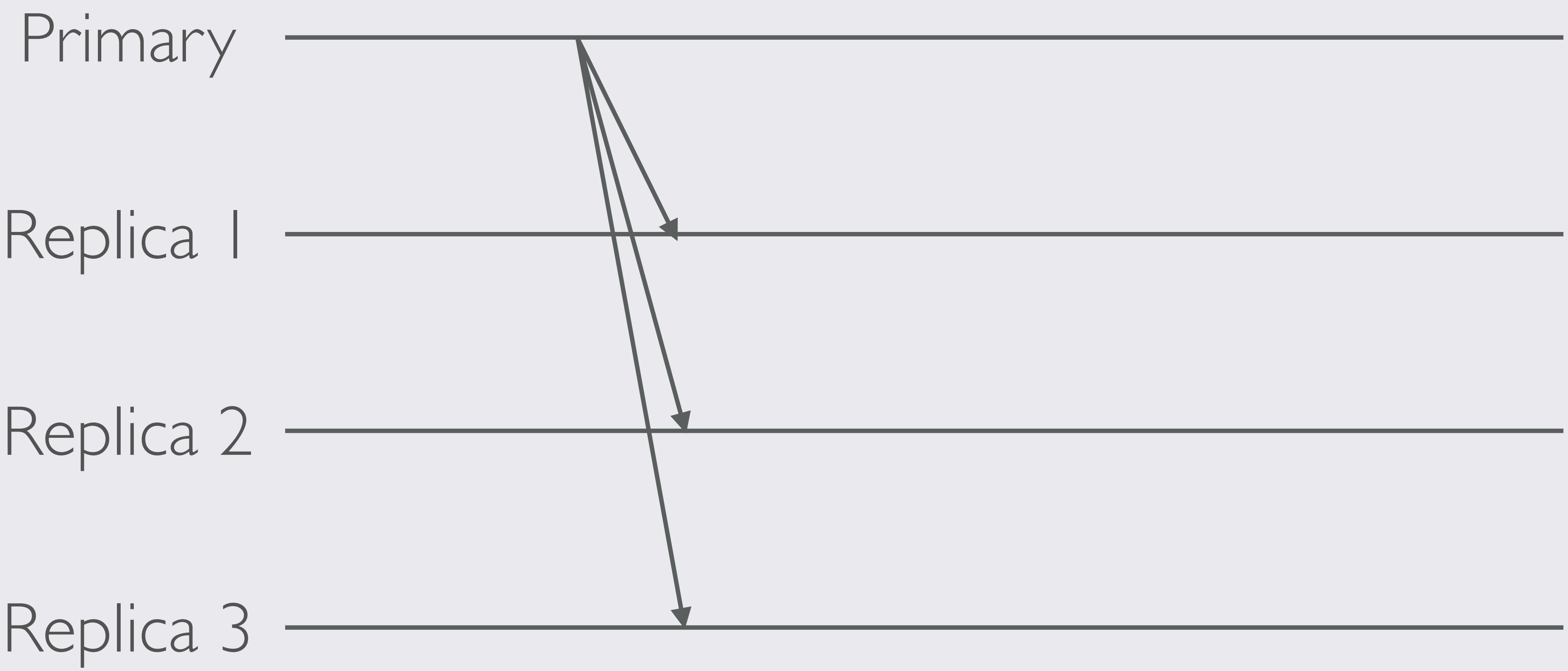
Replica 3

PRE-PREPARE

sequence number



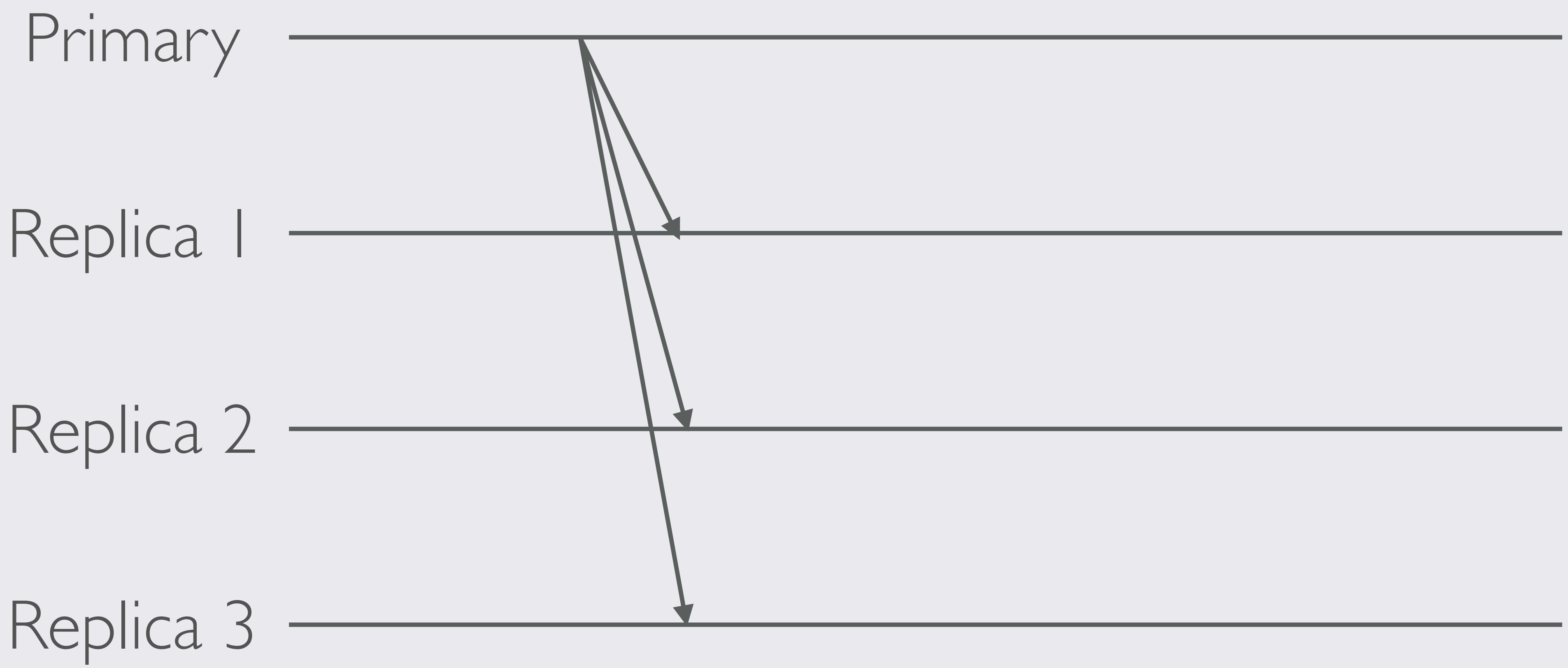
Primary sends <<PRE-PREPARE, v, **n**, d>$_{\sigma_p}$, m> to all replicas



PRE-PREPARE

client request

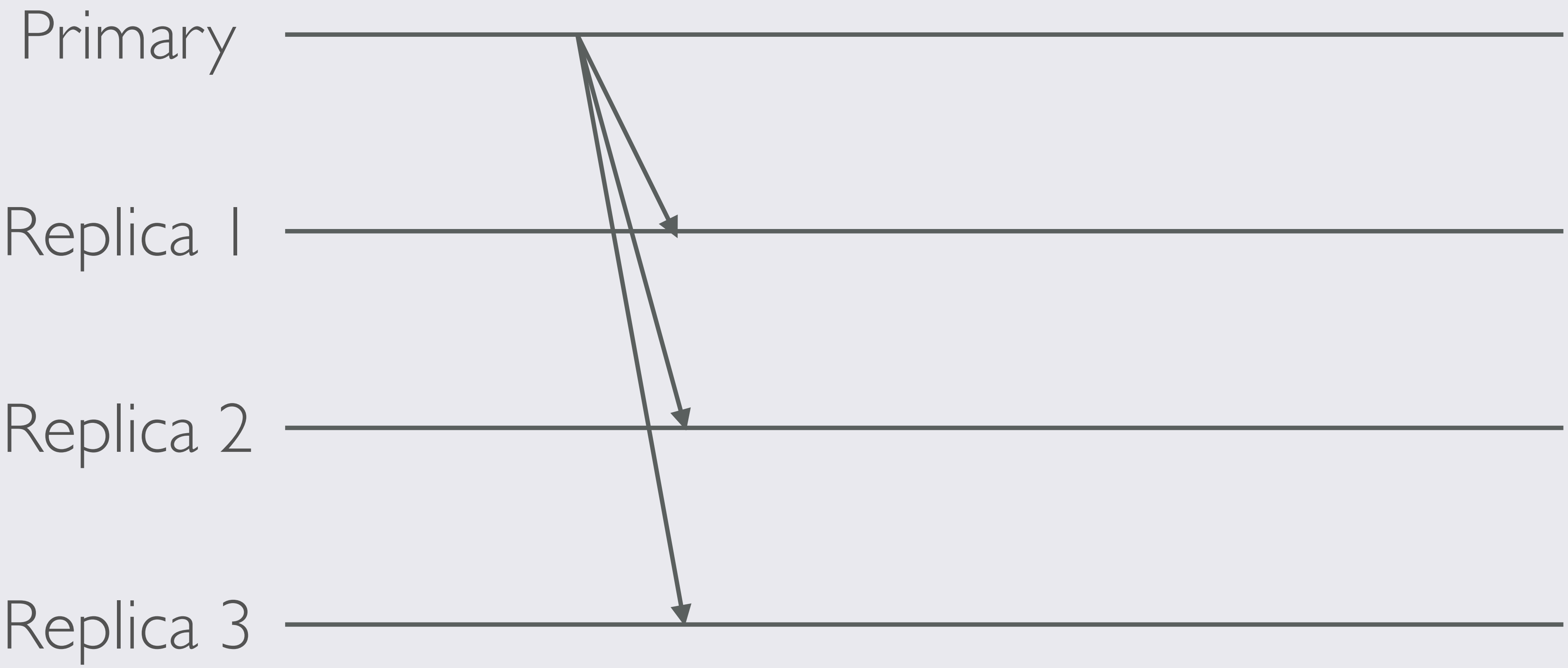
Primary sends <<PRE-PREPARE, v, n, d>$_{\sigma_p}$, **m**> to all replicas



PRE-PREPARE

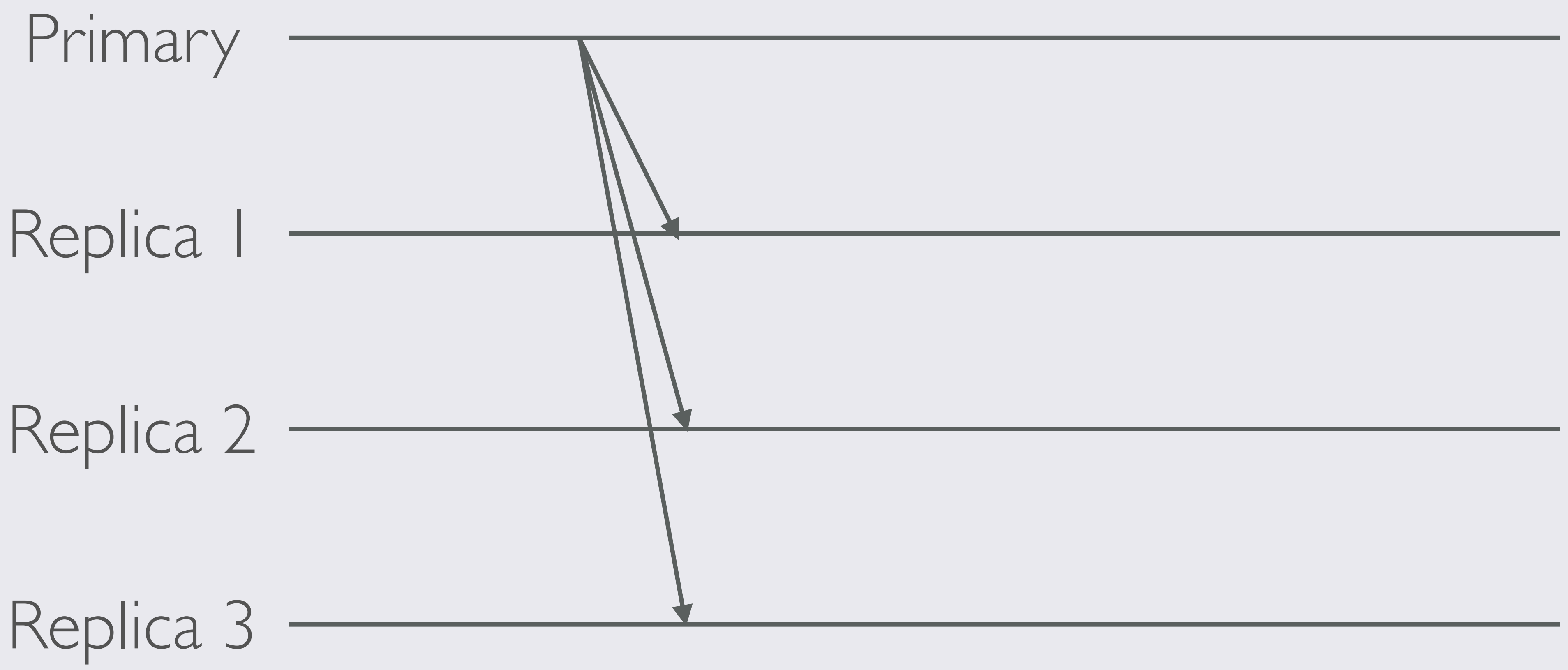
digest of m

Primary sends <<PRE-PREPARE, v, n, **d**>$_{\sigma_p}$, m> to all replicas



PRE-PREPARE

Primary sends <<PRE-PREPARE, v, n, d>$_{\sigma_p}$, m> to all replicas

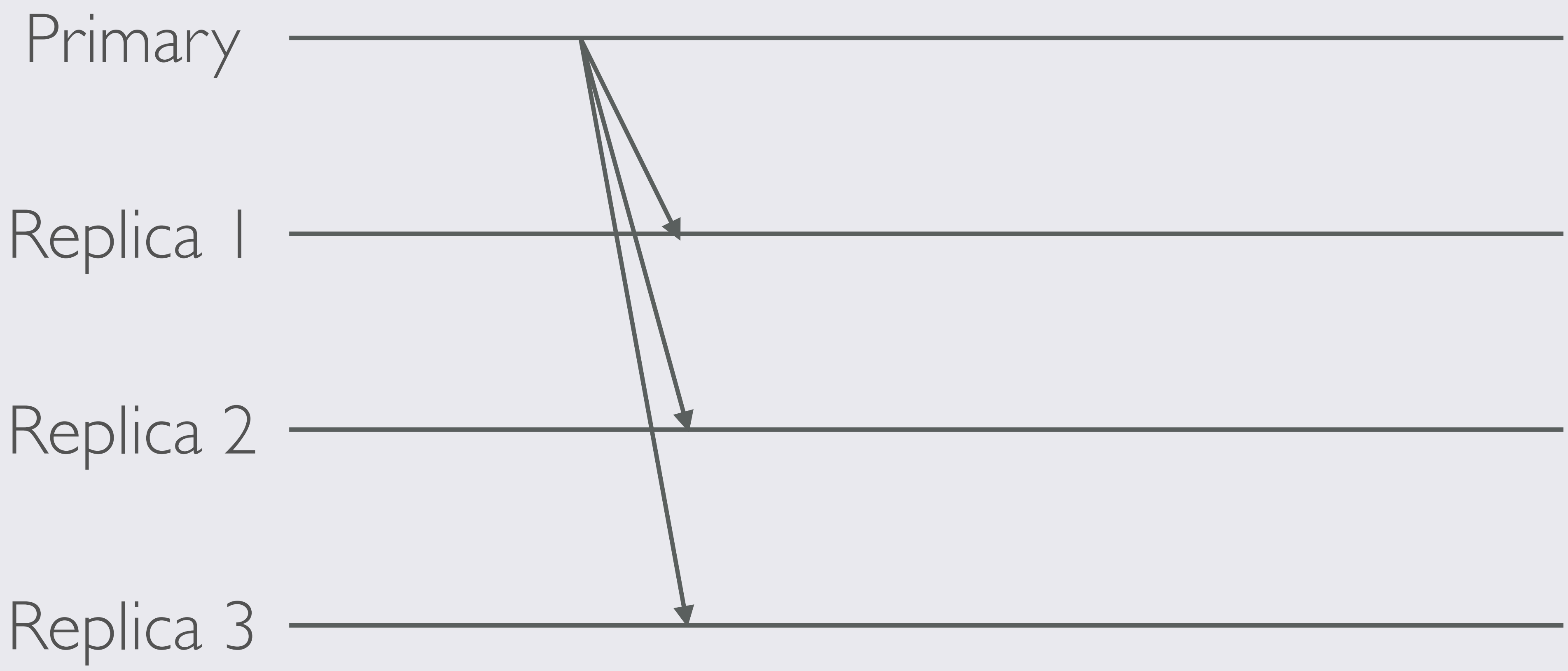


Correct backup **k** accepts PRE-PREPARE if:

- message is well formed
- **k** is in view **v**
- **k** has not accepted another PRE-PREPARE message for **v, n** with a different **d**
- **n** is between two watermarks **L** and **H** (to prevent sequence number exhaustion)

PRE-PREPARE

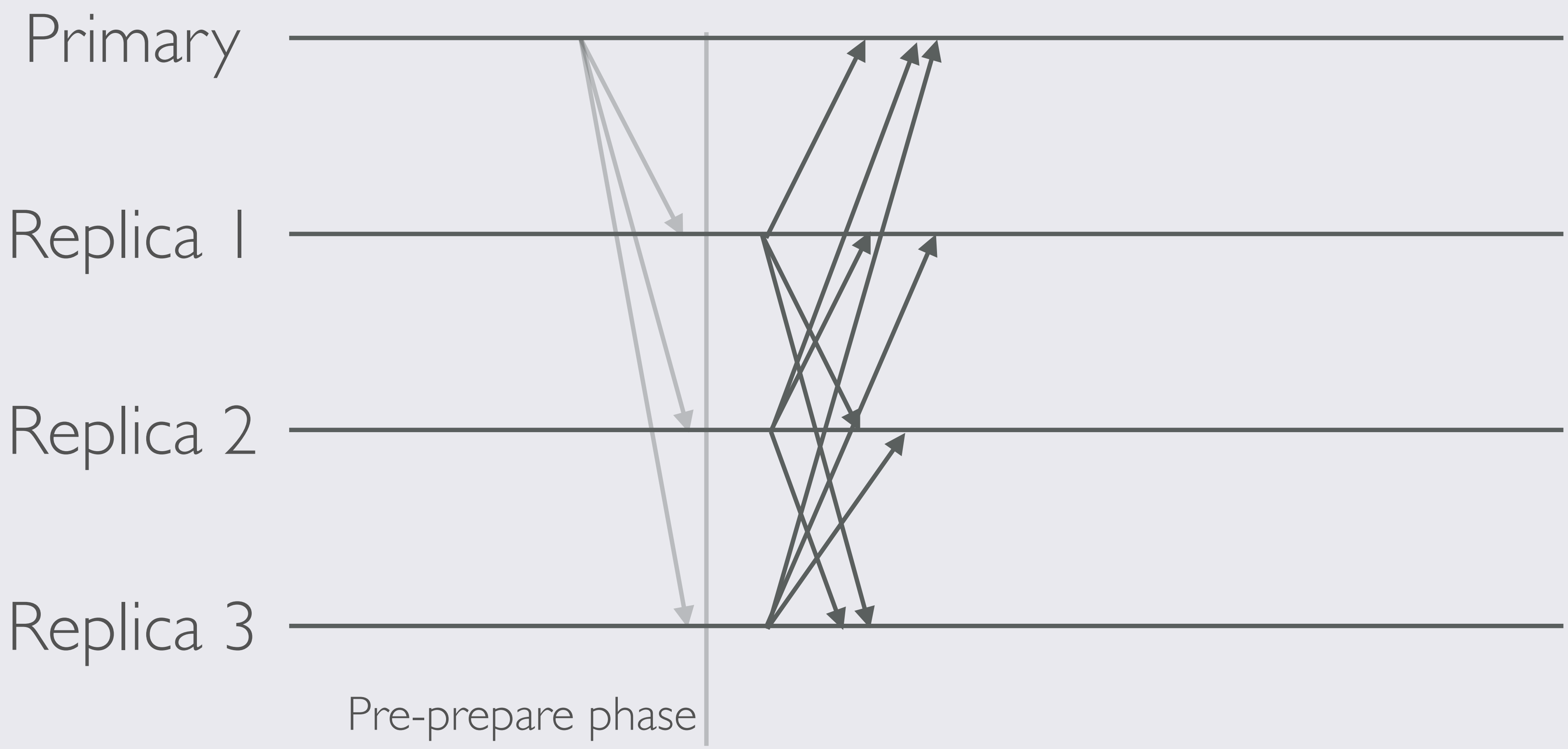
Primary sends <<PRE-PREPARE, v, n, d>$_{\sigma_p}$, m> to all replicas



Each accepted PRE-PREPARE message is stored in the accepting replica's message log (including the primary's)

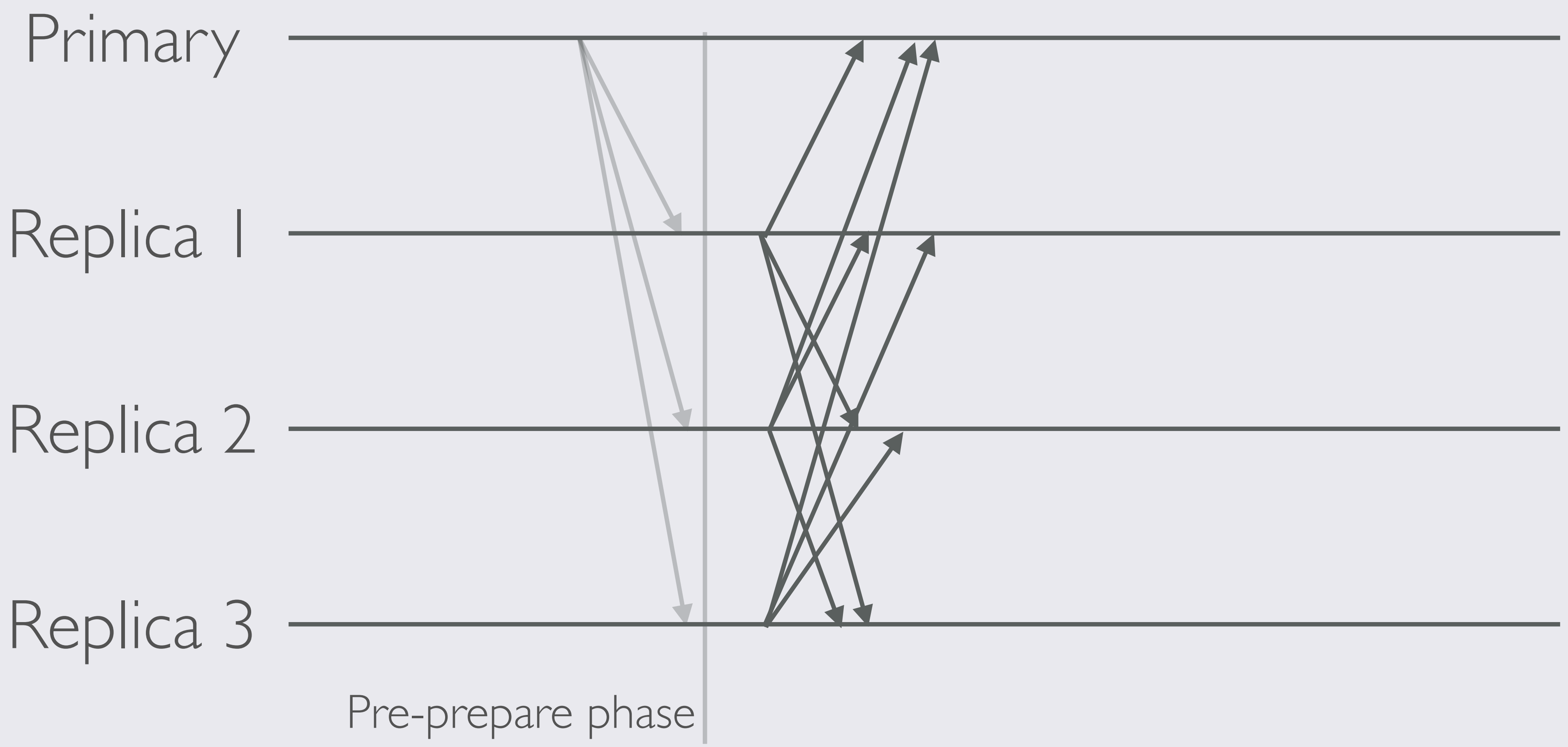
PREPARE

Replica **k** sends <PREPARE, v, n, d, k>$_{\sigma_k}$ to all replicas



PREPARE

Replica **k** sends <PREPARE, v, n, d, k>$_{\sigma_k}$ to all replicas

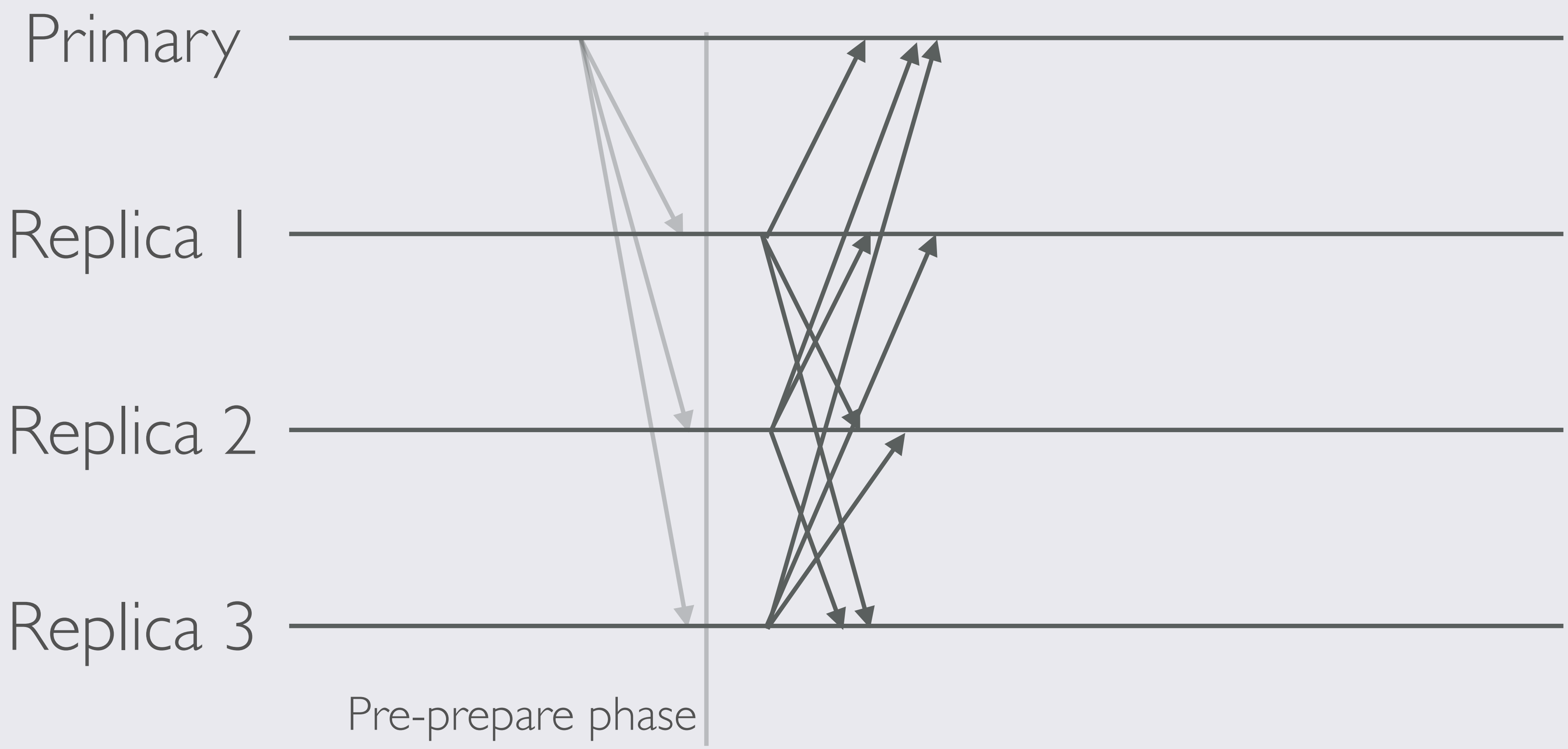


Correct backup **k** accepts PREPARE if:

- message is well formed
- **k** is in view **v**
- **n** is between two watermarks **L** and **H**

PREPARE

Replica **k** sends <PREPARE, v, n, d, k>$_{\sigma_k}$ to all replicas



- Replicas that send a PREPARE accept the assignment of **m** to sequence number **n** in view **v**
- Each accepted PREPARE message is stored in the accepting replica's message log

Prepare Certificate

- P-Certificates ensure consistent order of requests within views

- A replica produces a P-Certificate(**m**,**v**,**n**) iff its log holds:
  - the request **m**
  - A PRE-PREPARE for **m** in view **v** with sequence number **n**
  - $2f$ PREPARE from distinct backups that match the PRE-PREPARE

- A P-Certificate(**m**,**v**,**n**) means that a quorum agrees to assign **m** to sequence number **n** in view **v**
  - **No** two non-faulty replicas with P-Certificate(**m**,**v**,**n**) and P-Certificate(**m'**,**v**,**n**)

ADMINISTRIVIA

No class the next two Mondays

- Monday 10/18, UM study day
- Monday 10/15, conflict with SOSP workshops

Research part

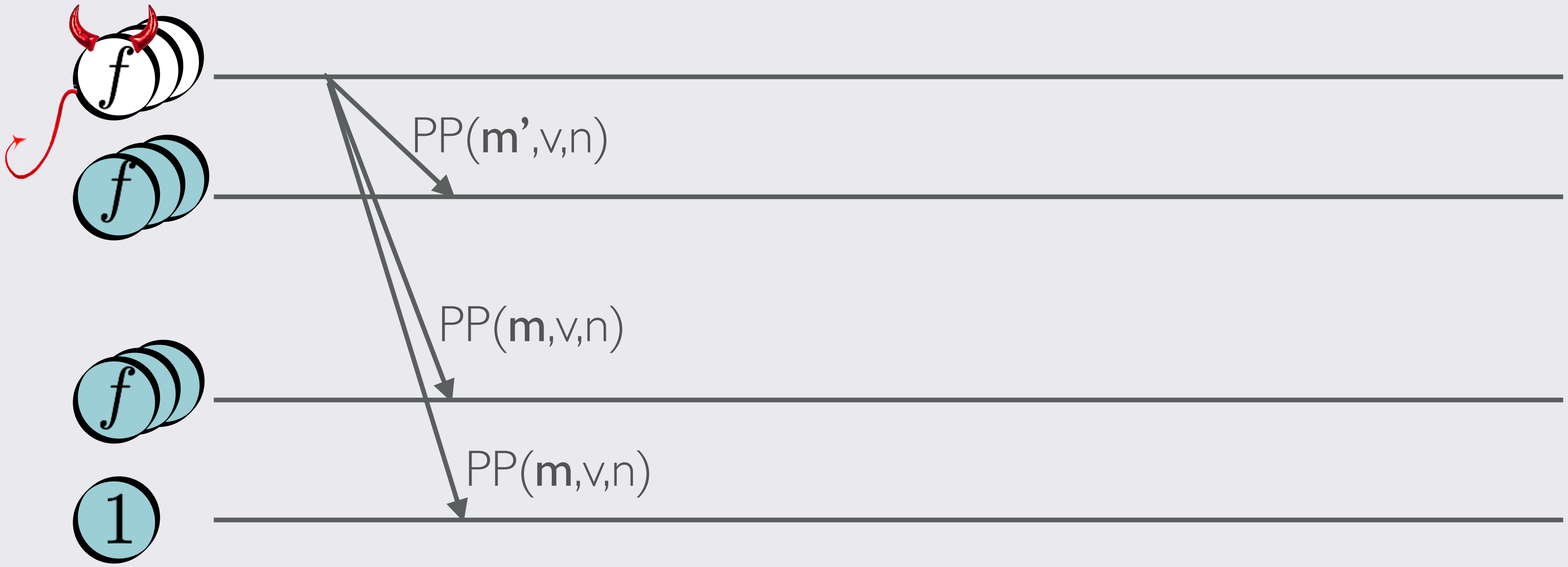
- Presentation schedule posted on class website
- Review submission website coming up around 10/25

P-Certificates are not enough

- A P-Certificate proves that a quorum of $2f+1$ replicas has agreed to assign **m** to sequence number **n** in view **v**
- Yet that assignment could be modified if a **view change** happens (the primary changes)
    - The new primary may not be convinced to assign **m** to **n** in the new view **v'**

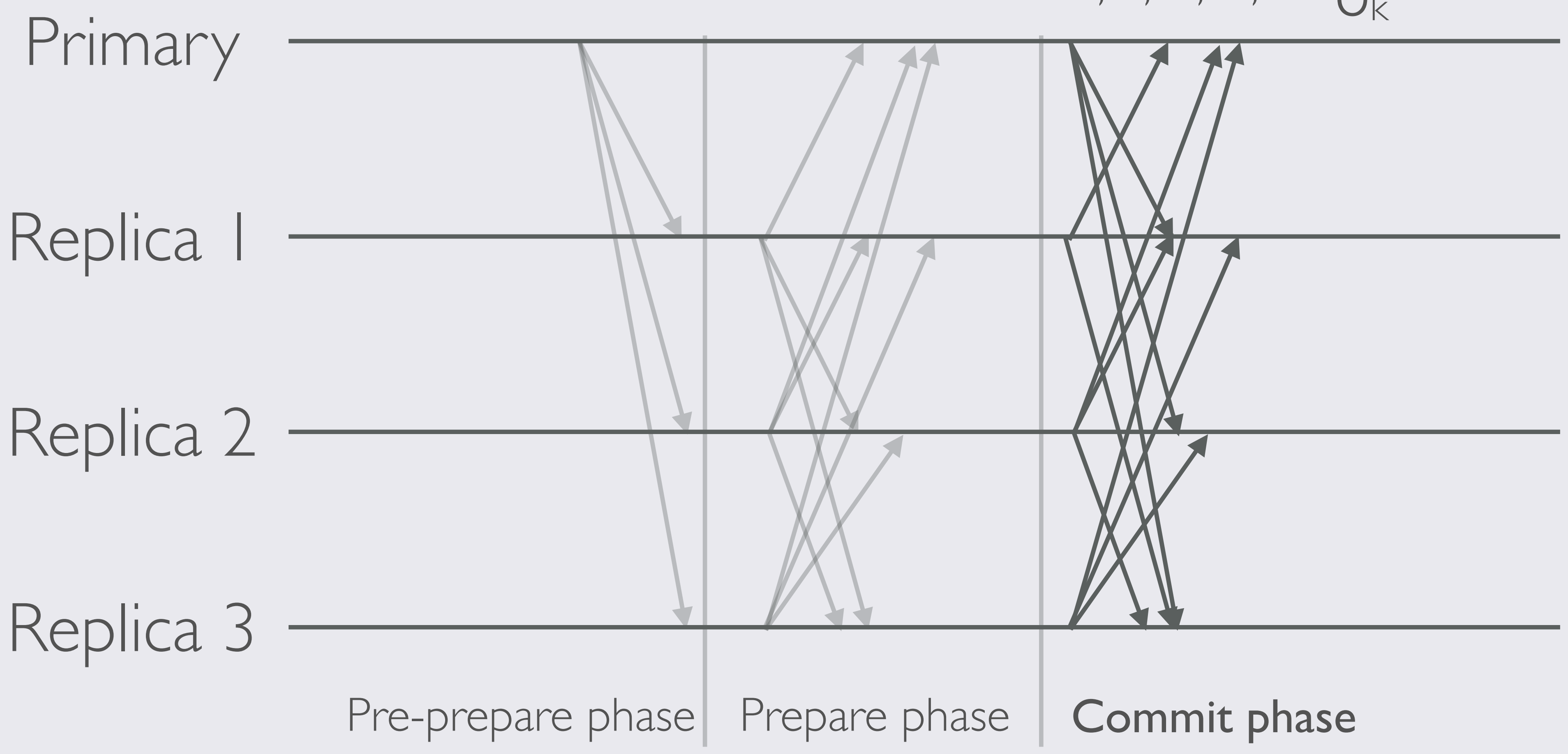
P-CERTIFICATES ARE NOT ENOUGH

- Yet that assignment could be modified if a **view change** happens (the primary changes)
  - The new primary may not be convinced to assign **m** to **n** in the new view **v'**
  - $2f+1$ prepares means at least $f+1$ correct replicas received a pre-prepare for (**m**,**v**,**n**)



COMMIT

After collecting a P-Certificate, replica k sends <COMMIT, v, n, d, k>$_{\sigma_k}$ to all replicas



COMMIT CERTIFICATE

- C-Certificates ensure consistent order of requests **across** views
  - **Cannot miss** a P-Certificate during view change

- A replica has a C-Certificate(**m**,**v**,**n**) iff:
  - it had a P-Certificate(**m**,**v**,**n**)
  - its log contains $2f + 1$ matching COMMIT messages from distinct replicas (including itself)

- A replica executes a request when:
  - it gets a C-Certificate for it
  - it has executed all requests with smaller sequence numbers